Unit Test

Walter Zhou
2015-03-03
walterzh@marvell.com

unit test function

```
typedef int (*utest_t)();
```

For example:

```
   int cpu_test(void);
```

return 0, test successful

return !0, test failure

unit test sample

```
int cpu_test(void)
{
    ……  // unit test code
}

UNIT_TEST(“cpu”, cpu_test);
```

run the unit test

CMD==> test run cpu cpu_test

run_utest_task started. group=cpu name=cpu_test

RUN TEST: cpu cpu_test:

CMD==> CPU UT: Testing interrupt state change...passed.
CPU UT: Testing get cycle count...
Count0: 2070321761
Count1: 2070321842
Count2: 2070321926
passed.
CPU UT: Testing cpu get mode...passed.
PASSED: cpu cpu_test
TEST PASSED: 1

unit test command

CMD==> test run **group_name func_name**

UNIT_TEST(“cpu”, cpu_test);

“cpu” is group name

cpu_test is function name

unit test command

CMD==> test run **group_name**

To run all function tests in the group.

For example:

UNIT_TEST("rtc", rtc_test_each_bit);

UNIT_TEST("rtc", testRTCGet);

UNIT_TEST("rtc", testRTCGetSet);

unit test command

CMD==> test run

To run all test cases.

CMD==> test list

List all test cases

list all test cases

CMD==> test list
UTEST[util] testEndian
UTEST[util] testDelta
UTEST[date_time] date_time_test
UTEST[log] error_log_test
UTEST[gloss] gloss_test_gettimeofday
UTEST[gloss] gloss_test_printf
UTEST[gloss] gloss_test_file
UTEST[oid] oid_unit_test
UTEST[paper] paper_tests
UTEST[cdma] cdma_test
UTEST[cpu] cpu_test
UTEST[gpio] gpio_unit_test
UTEST[jbig] jbig_test
UTEST[nand] nand_test
UTEST[rtc] testRTCGetTime
UTEST[rtc] rtc_test_each_bit
UTEST[rtc] testRTCGet
UTEST[rtc] testRTCGetSet
UTEST[rtc] testRTCAlarmSet
UTEST[rtc] testRTCAddTime
UTEST[rtc] testRTCCalGetSet
UTEST[timer] timer_test
UTEST[uart] testUARTExecTestSuite
UTEST[usbdev] usb_device_test
UTEST[spi] spi_unit_test
UTEST[strfmt] strfmt_test

[util] is group name

testEndian is function name

unit test coding rule

- Rule - 1

If possible, every implementation file(.c) **needs** write unit test case(s); if not, please **explain** why you could skip unit test when code review.

- Rule - 2

If add new feature to the existing implementation file(.c), the developer **needs** add unit test case(s) for add / modify function(s)

Where to write unit test case

- Rule - 3

Unit test case could be written in the implementation file or in a new file. If in the same file with the implementation one, please make unit test function protect by HAVE_UNIT_TEST. For example,

```
#ifdef HAVE_UNIT_TEST
    int cpu_test()
    {
      ……
    }
#endif
UNIT_TEST(“cpu”, cpu_test);
```

Where to write unit test case

- Rule - 4

If in a new file, please exclude unit test file(s) in release version. For example,

In makefile

SOURCE += ipp.c

SOURCE += ipp_attribute.c

ifdef HAVE_UNIT_TEST

SOURCE += ipp_attribute_test.c

Endif

SOURCE += ipp_const.c

One task at a time

Rule - 5

```
/// modifies width for best fit with video.
/// returns scale factor or 0 on error
static uint32_t urf_papersize_adjust( urf_page_t *urf_page, uint32_t *x_offset )
{
    uint32_t scale = 0;
    uint32_t width_max = 0;

    // Get the scale factor for sub 600 dpi resolutions
    scale = scale_from_resolution(urf_page->resolution);

    if (scale)
    {
        width_max = MAX_WIDTH_IN_PIXELS / scale ;  // approx 100 pixel margin

        // set width to data width modulo BUFFER_MOD and offset to zero let video center
        width_max = urf_page->width / BUFFER_MOD;
        width_max *= BUFFER_MOD;

        *x_offset = (urf_page->width - width_max) / 2 ;
        ……
}
```

One task at a time

```
if (!urf_papersize_adjust(&urf_page,
     scale = scale_from_resolution(urf_page.resolution)))
{
    break; // error handling
}
```

Naming Test Functions

Rule – 6

Name test function friendly

```
void SortAndFilterDocs_test()
{
    …
}
void Test1_test() --- Bad name
{
    …
}
```

Small Function

Rule – 7

Create new function as small as possible.

It is difficult to test for big function, and giant function is almost not testable.

The coding standard define the preferable code line in a function.

Don’t access global variable in function

Rule – 8

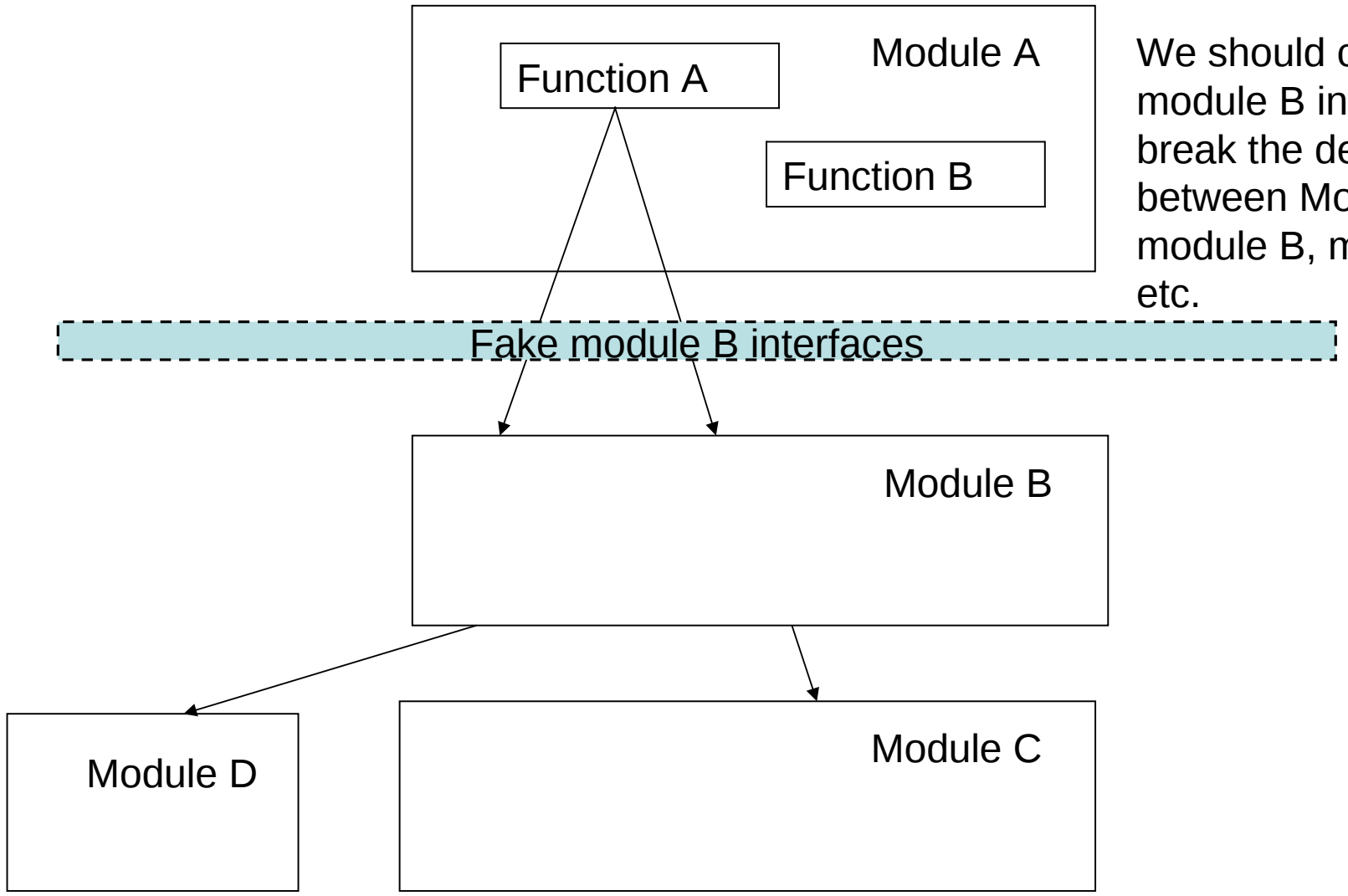
If you need access global variable(s), pass them by function parameter(s).

Break dependencies

Rule - 9

To design more modular and testable C, we employ a header file to publish the interface of a module. A testable module is one that interacts with other modules through the module’s interface.

Break dependencies



We should create “fake module B interfaces” to break the dependency between Module A and module B, module C, etc.